EXHIBIT 1
FILED UNDER SEAL

DECLARATION OF SAMRAT BHATTACHARJEE

I, Samrat Bhattacharjee, declare as follows:

1. I previously submitted a declaration in support of Google's Motion for Summary Judgment Regarding Claim 13 of U.S. Patent No. 9,967,615 against Sonos, Inc. ("Opening Declaration"). I submit this declaration in further support of Google's motion.

2. Sonos's expert, Dr. Schmidt, opines that the "plain and ordinary meaning of the term 'local playback queue'" is "a data construct on the playback device that can contain one or more resource locators (*e.g.*, videoIds, URLs, or other resource locators), where each resource locator corresponds to multimedia content (*e.g.*, a particular song or video) that the playback device is to playback." Schmidt Decl., ¶69. I disagree that Dr. Schmidt's characterization is the plain and ordinary meaning of the term "local playback queue," as claimed. As I showed in my opening report, "[a] POSITA would understand that a typical implementation of a playback queue links together different multimedia items in a particular order using linked lists, arrays, vectors, or other well-known data structures" and "any implementation of a queue would allow for the storage of multiple items." Opening Decl., ¶70. "A POSITA would not consider a fixed set of named variables to be a playback 'queue.'" *Id.*

3. After I submitted my original declaration, Sonos produced printouts of source code relating to its prior art zone players, which the '615 patent identifies as exemplary "playback devices." '615 patent at 4:26-53; Figs. 4, 6. The playback queue implementation in Sonos's zone players further supports my opinion because Sonos's zone players use a playback "queue" implementation that is consistent with how that term is understood in the art. Specifically, the playback queue data structure in Sonos's zone player is defined and implemented by the files tqueue.h and tqueue.cxx. In particular, the playback queue is stored as an array that can be populated with a variable number of tracks. The playback queue also supports queue management, such as the ability to add and remove tracks, to query for the next and previous track, reorder tracks, and to clear the playback queue by deleting all tracks. The

playback queue is not implemented as a fixed set of named variables. *See* tqueue.h and tqueue.cxx.

4. In support of Sonos's infringement arguments, Dr. Schmidt opines that an "intermediate service (the Player Service) [] translates [a] videoId into at least one URL for the particular media item at a YouTube Bandaid server." Schmidt Decl., ¶119. Dr. Schmidt does not cite to any evidence for his assertion that the Player Service "translates [a] videoId into at least one URL," and to the extent Dr. Schmidt is suggesting that there is a direct mapping of a videoId to a URL, I disagree. A videoId is input to a "Mapping Service" that uses other information—including the viewing device's IP address, network conditions, and various other options to generate a URL that identifies a Bandaid server from which the receiver should request the content. Notably, the Mapping Service uses factors, such as the client IP address and network conditions, that are entirely unrelated to the videoId, which reinforces my view that the videoId is not "translated" into the URL. And because the Mapping Service uses dynamic factors, the same videoId can result in different URLs for each user or for the same user each time the video is played back, as I explained in my Opening Report. Opening Report, ¶97.

5. Dr. Schmidt also opines that the receiver adds "one or more URLs for each video to the Receiver's memory" in a "DashManifest." Schmidt Decl., ¶120. As I explained in my Opening Declaration, the DashManifest is not added to the variables or WatchNextResponse message that Sonos identified as the alleged local playback queue. Opening Decl., ¶98. Dr. Schmidt does not appear to dispute this, but opines that a DashManifest is its own playback queue because it is a type of "data construct." I disagree. A person of skill in the art would not consider a DashManifest to be a playback queue. A DashManifest includes information for a single video. Specifically, a DashManifest is a standard format used in the industry with the MPEG DASH video streaming protocol. When a video in the DASH format is uploaded to a video streaming service like YouTube, it is broken up into pieces (called "chunks") and a DashManifest file is created that includes the information necessary to recreate the video. This information includes URLs that point to the relevant chunks for a *single* video. In other words,

the DashManifest does not provide information about a playback queue—but instead a single media item that may be part of the playback queue. In contrast, any implementation of a playback queue would allow for the storage of multiple media items (i.e., the ability to queue up items). A playback queue would also allow for queue management functions (e.g., adding, removing, or reordering items in the queue), and Dr. Schmidt has not (and cannot) show that the DashManifest supports queue management functions.

6. Dr. Schmidt also opines that the "resource locator" limitation is met under the doctrine of equivalents. Schmidt Decl., ¶122. I disagree. In the claimed system, the instruction to transfer playback causes a set of cloud servers to add one or more resource locators that point to the location of the multimedia content to the playback device. Unlike the claims, in the accused system, the instruction that causes playback to be transferred does not cause one or more resource locators to be added to the playback device. At most, it causes a videoId to be added to the playback device. The URLs are added one-by-one later by using a separate process that, among other things, maps the playback device to the appropriate content server based on factors such as network latency, system load, and policy factors. Performing these functions at the time each video is played back enables optimizations (*e.g.*, taking into account the user's current location and load on the content delivery network) that are not disclosed in the '615 patent, and necessary for a system that operates at the scale of YouTube. Moreover, YouTube breaks the multimedia content into pieces (called chunks) and may use different resource locators for different chunks of the media item or formats, rather than a single resource locator that points to the entirety of the content. Thus, Google does not infringe under the DOE at least because it does not perform substantially the same function (*e.g.*, at least because the original function in the claim receives the one or more URLs for the multimedia content in the playback queue as part of the step of sending playback from the sender to the receiver, but the accused system receives URLs one-by-one at the time the multimedia content is to be played back on the receiver), in substantially the same way (*e.g.*, at least because the accused systems receives URLs to one of many possible servers after performing a mapping that takes into consideration

various network conditions at the time of playback), to achieve substantially the same result (*e.g.*, at least because in the accused systems there is no one URL that can be considered a single resource locator for the multimedia content).

7. Dr. Schmidt further opines that adding "multimedia content" and "resource locators" to the playback queue would introduce "redundancy" in the claim language. Schmidt Decl., ¶114. I disagree. A POSITA would understand that a playback device could store the multimedia content and resource locators. Such a system would allow the playback device to render the content when the device is offline. It would further enable flexibility by permitting the playback device to delete the content after a specified amount of time (for instance, due to restrictions placed on the content through digital rights management or memory capacity constraints) and re-fetching the content using the resource locator, as necessary, upon demand.

8. I have also reviewed the CloudQueueSyncCoordinator.java file that Dr. Schmidt cites to in his declaration with respect to the GPM product. Schmidt Decl., ¶55. Dr. Schmidt cites a comment in the code that references a receiver refreshing its queue. However, Dr. Schmidt has not cited any source code on the receiver that retrieves a copy of the cloud queue and stores it at the receiver. And the file CloudQueueSyncCoordinator.java does not even run on a receiver—it runs on a sender device.

9. I have confirmed that YouTube and Google Play Music allow users to create playlists of variable size that may include hundreds or thousands of media items. I have not seen any part of the source code that restricts the size of the playback queue to three media items, whereas the alleged "local playback queues" that Dr. Schmidt has identified are restricted to at most three items.

I, Samrat Bhattacharjee, declare under penalty of perjury under the laws of the United States that the foregoing is true and correct.

Dated: May 19, 2022

[illegible]

Samrat Bhattacharjee